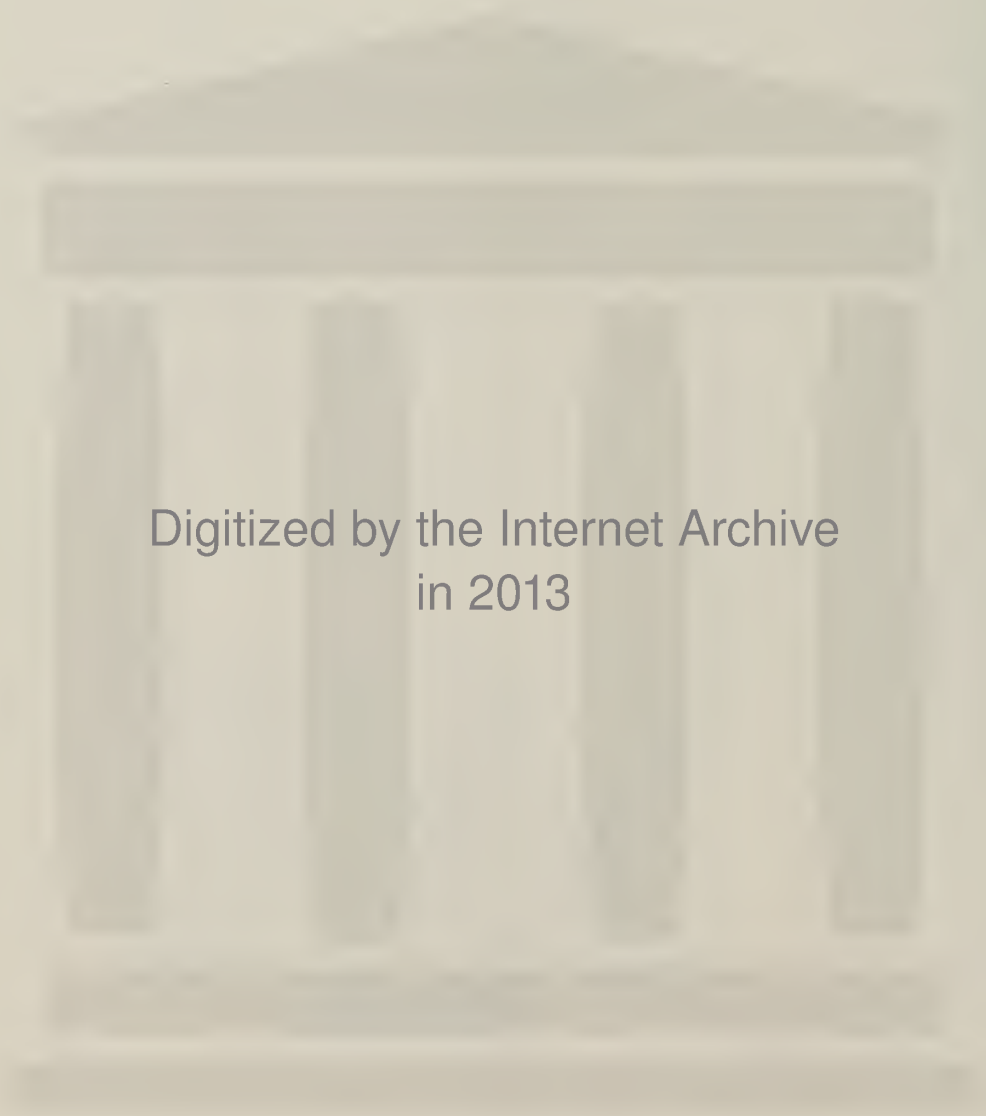


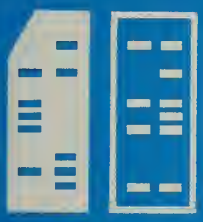
Report No. 436

EMULATION OF DISC FILE PROCESSOR

by

Hirohide Yamada

June 1971



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. 436

EMULATION OF DISC FILE PROCESSOR

by

Hirohide Yamada

June 1971

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

*This work was submitted in partial fulfillment of the requirements for the degree of Master in Science in Computer Science, 1971.

Acknowledgement

The author would like to express his great appreciation to his advisor, Dr. D. Kuck, for his sincere guidance throughout this work.

The author also would like to convey his sincere gratitude to Mr. D. H. Lawrie for his guidance.

Thanks are also extended to Mr. C. E. Carter, Mr. Bill Stellhorn and Mrs. Sandy Moy for their help.

Table of Contents

		Page
I.	Introduction	1
II.	Register Configuration	4
III.	Definition of Word Instruction	5
IV.	Example of Word Instruction	9
V.	Definition of String Instruction	11
	A. S1 format	11
	B. S2 format	13
	C. S3 format	14
	D. S4 format	16
	E. Decoding sequence	18
	F. String instruction specification	19
	1. Search forward	19
	2. Search reverse	20
	3. Compare	21
	4. Find	23
	5. Move forward	24
	6. Move reverse	25
	7. Translate	26
VI.	Example of String Instruction	27
List of References		32
Appendix		33

I. Introduction

Today many big and fast computers are built, and in those computers, central processors are sophisticated so as to handle a variety of jobs in a fast way. On the other hand, in this real world, there are many jobs which have a lot of data but need only a little bit of simple data manipulation and the main task is to look for certain data that you want. For example, you digitize magazines of your favour and keep them on disk, and if you want to look at a certain article, you give a computer certain keys which indicate the article and the computer displays the article. In this kind of job, the main task of the central processor is searching the keys in the data stream.

Our objective is to build a data processing system which handles simple operations, mainly search, on large amounts of data within a reasonable time and very cheaply.

The outlook of the system is shown in Figure 1. All the data is stored in disks and is brought to S memory in blocks and processed. S memory is the main memory and contains programs for processing the data. Those instructions in S memory, which are called S-instructions, are fetched one by one by the interpreter and interpreted

by the program in MN memory. S instructions are categorized into three parts, namely Word instruction, String instruction and I/O instruction. In this paper the definition of Word instruction and String instruction and their interpretation are presented.

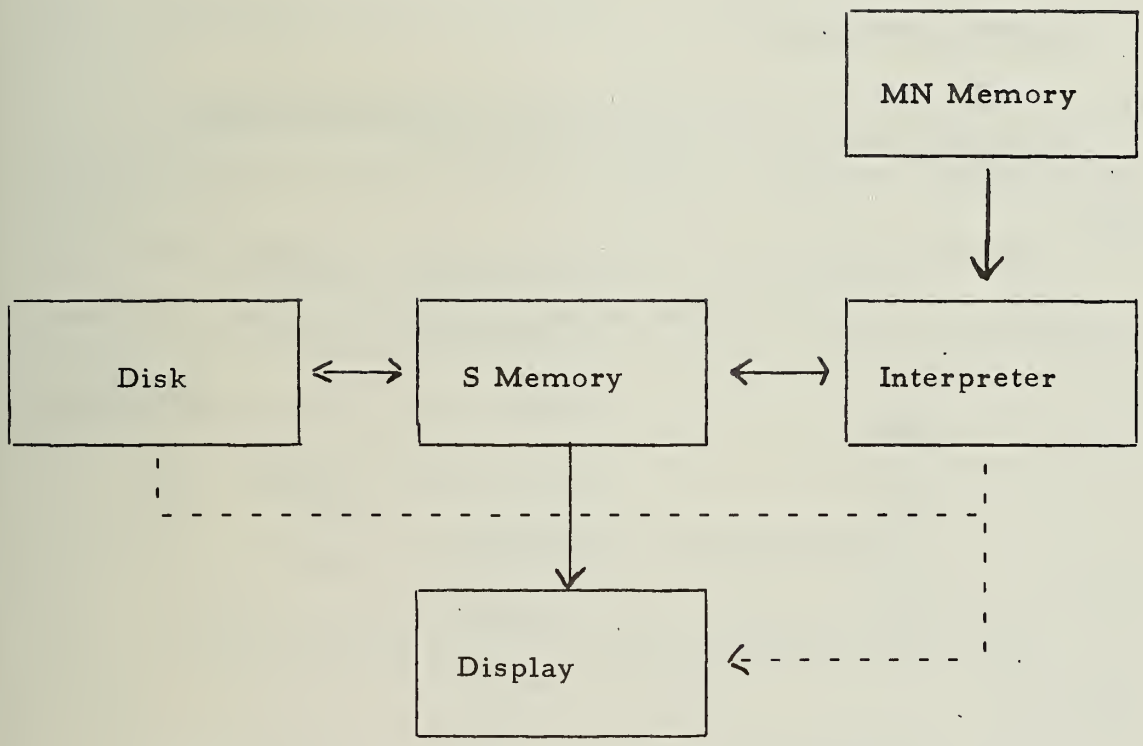


Figure 1 System Configuration

—— data line

- - - control line

II. Register Configuration

The interpreter has the following registers in the logic unit:

A1, A2, A3	16 bit general register
B	16 bit buffer register
MIR	16 bit memory information register
CTR	8 bit counter register
LIT	8 bit literal register

Above those registers there are following registers in S memory:

Register	Memory Address	Explanation
IAR 0 — 15	16 — 31	16 bit information address register 0 — 15
PTR 0 — 15	32 — 47	16 bit pointer register 0 — 15
CHA 0 — 15	48 — 63	8 bit character register 0 — 15 only least significant byte of a memory word is used
CTR 0 — 15	64 — 79	8 bit counter register 0 — 15 only least significant byte of a memory word is used

III. Definition of Word Instruction

A word instruction consists of OP code, X1, X2, and X3, where each takes one S memory word as shown in Figure 2. The format of OP code is shown in Figure 3. OP code is decoded from least significant bit to most significant bit and the meaning of each bit is as follows:

Bit	
15	0 indicates word instruction
13, 14	Indicates level of indirectness for the first operand as follows;
	00 X1 is the first operand
	01 X1 addresses the first operand
	10 X1 indirectly addresses the first operand
	11 X1 doubly indirectly addresses the first operand
11, 12	Same as bit 13, 14 except for the second operand
4 10	According to the value of this field, the following action is taken;
	0 Operand1 and operand2 = A1
	1 Operand1 and operand2 = A1

Figure 2 Word instruction format

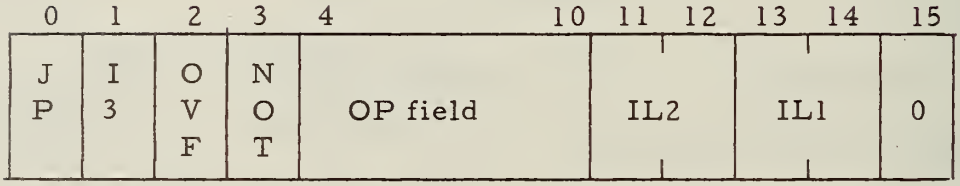


Figure 3 OP code format

2 Operand1 eqv operand2 = A1 In this operation A1 bits are set to one if the corresponding bits of both operand1 and operand2 are the same. Otherwise A1 bits are reset.

3 Operand1 is shifted right by the amount of operand2 and placed in A1

4 Operand1 is shifted left by the amount of operand2 and placed in A1

5 Operand1 is rotated right by the amount of operand2 and placed in A1

6 Operand1 + operand2 = A1 If overflow does not occur, OVF bit is reset

7 Operand1 — operand2 = A1 If overflow does not occur, OVF bit is reset

8 Operand1 is compared with operand2 and if those are the same, 8000 (hex) is set to the A1. Otherwise A1 is reset.

9 Address of X2 of current instruction plus operand2 is stored in address specified by operand1, and A1 is set to 8000 (hex)

3 If this bit is on, A1 is complemented

2 If this bit is on, the address of current instruction minus one is stored in LAR0 and program control is passed to the content of IAR1 plus one. I3 and JP bits are ignored.

1 If this bit is off, X3 is passed as the address to bit 0. If on, content of S memory addressed by X3 is passed to bit 0 as the address.

0 If this bit is off, A1 is stored into the S memory address passed from bit 1 and control goes to the next S instruction. If on, and sign bit of A1 is on, program control is passed to the instruction whose first word address is the address passed by bit 1 plus one. If this bit is on, and sign bit of A1 is off, program control is passed to the next S memory instruction.

IV. Example of Word Instruction

The following notations are used to ease the explanation:

X	Literal X or address X
(X)	Content of S memory word at address X
((X))	Content of S memory word at address (X)
X	Store into address X
d	Don't care

Example 1 X1 OR (X2) = X3

JP	I3	OVF	NOT	OP field	IL2	IL1	Bit 15
0	0	0	0	1	01	00	0

Example 2 If sign bit of (X1) and (X2) are not the same, jump to X3. Else go to next instruction.

JP	I3	OVF	NOT	OP field	IL2	IL1	
1	0	0	1	2	01	01	0

Example 3 (X1) + ((X2)) and if overflow does not occur in this operation, store it into (X3). If overflow occurs, don't change (X3), save current instruction address into IAR0 and jump to (IAR1).

JP	I3	OVF	NOT	OP field	IL2	IL1	
0	1	1	0	6	10	01	0

Example 4 If (X1) and X2 are the same jump to X3+1, else go to next.

JP	I3	OVF	NOT	OP field	IL2	IL1	
1	0	0	0	8	00	01	0

Example 5 Store current address + X2 into X1 and jump to X3.

JP	I3	OVF	NOT	OP field	IL2	IL1	
1	0	0	0	9	00	00	0

V. Definition of String Instruction

A string instruction consists of four consecutive words S1, S2, S3 and S4 in S memory as shown in Figure 4.

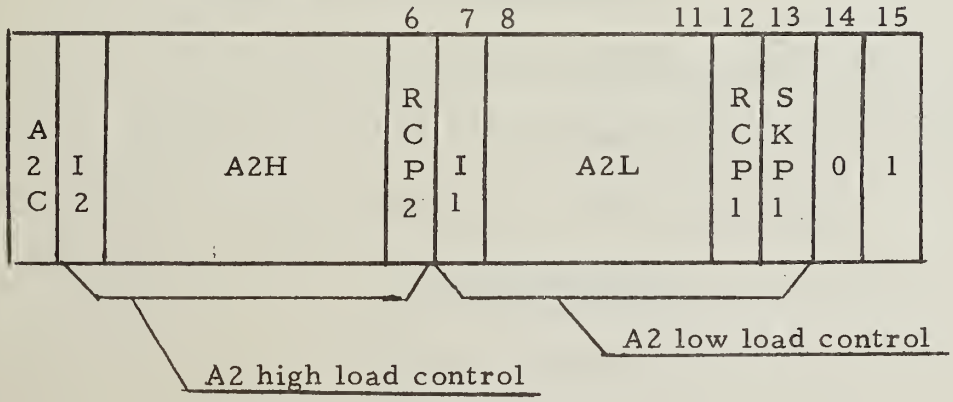
Figure 4 String instruction format

The detailed format of each word and its meaning is explained below. Each instruction is decoded from S1 to S4 and from the least significant bit to the most significant bit in each word.

A. S1 format

The detailed format of S1 word is shown in Figure 5, and the meaning of each bit is as follows;

Figure 5 S1 word format



Bit	
14-15	Each string instruction has always 01 in these bits.
13	If this bit is on, bits 7 – 12 are neglected. If this bit is off, bits 7 – 12 have the meaning as explained below.
12	If this bit is off A2L points to one of the character registers. If on, A2L points to one of the pointer registers.
8-11	These bits are a register number which is selected by bit 12.
7	If this bit is on, the lower byte of the A2 register is loaded by the character pointed by the content of the register specified by A2L and RCP1. If this bit is off, the lower byte of A2 register is loaded by the character in the lower byte of the register specified by A2L and RCP1.
1-6	Same as bit 7 – 12 except those bits load into higher byte of A2 register.
0	If this bit is on, bit 1 and bits 7 – 12 are all neglected and the full A2 register is loaded by the contents of the register specified by A2H and RCP2. In this case I2 must be zero. If this bit is off, this bit is neglected.

B. S2 format

The detailed format of S2 word is shown in Figure 6, and the meaning of each bit is as follows:

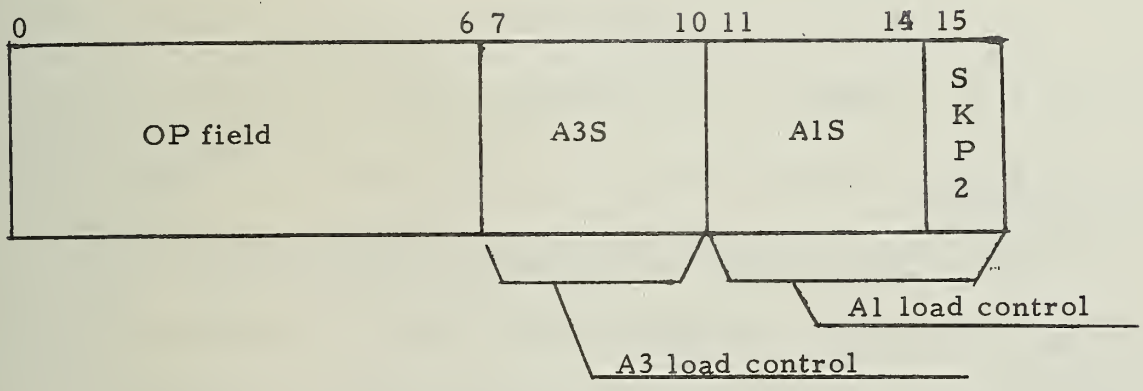


Figure 6 S2 word format

Bit

15 If this bit is on, A1 register is cleared and bits 11 — 14 are ignored. If this bit is off, A1 register is set according to A1S.

11-14 This is a character register number and upper byte and lower byte of A1 register are loaded by its content.

7-10 This is a pointer register number and A3 register is loaded by its content.

0-6 This field distinguishes the instructions as follows and the detailed explanation of each instruction is given in V.F.

0	Search forward
1	Search reverse
2	Compare

3	Find
4	Move forward
5	Move reverse
6	Translate
Others	Undefined

C. S3 format

The detailed format of the S3 word is shown in Figure 7, and the meaning of each bit is explained next. Bits 15-11 are decoded before the execution of the execution routines which are explained under F. Bit 10-0 are decoded after the execution of them.

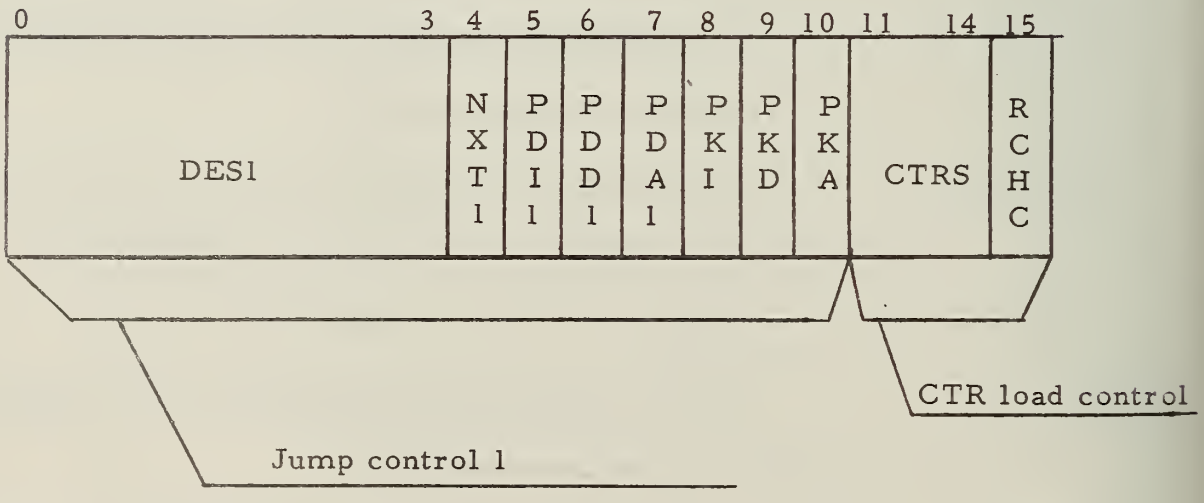


Figure 7 S3 word format

Bit	
15	If this bit is on, bits 11 – 14 indicate one of the counter registers (CTR0 – CTR15). If this bit is off, bits 11 – 14 indicate one of the character register (CHR0 – CHR15).
11-14	This field is a register number selected by bit 15 and the CTR register is loaded by the lower byte of its content.
10	If this bit is on the key pointer is incremented by one.
9	If this bit is on, the key pointer is decremented by one.
8	If this bit is on, the key pointer is initialized to the position before the execution of this String instruction.
7	If this bit is on, the data pointer is incremented by one.
6	If this bit is on, the data pointer is decremented by one.

5	If this bit is on, the data pointer is initialized to the position before the execution of this String instruction.
4	If this bit is on, program control is passed to the next instruction. If this bit is off, program control is passed to the instruction pointed by DES1. DES1 must point to the address to which you want to jump, minus one.
0-3	This field is an Instruction register number, and if bit 4 was off, program control is passed to the instruction specified by its content plus one.

D. S4 format

The detailed format of the S4 word is shown in Figure 8 and the meaning of each bit is as follows;

Bit	
15	If this bit is on, the data pointer is incremented by one.
14	If this bit is on, the data pointer is decremented by one.
13	If this bit is on, the data pointer is initialized to the position before the execution of this String instruction.

12 If this bit is on, the program control is passed to the next instruction. If this bit is off, program control is passed to the instruction specified by DES2.

8-11 This field is an Instruction address register number and if bit 12 was off, program control is passed to the instruction specified by its content, plus one.

0-7 Same as bit 8-15 except that these bits are for jump control 3.

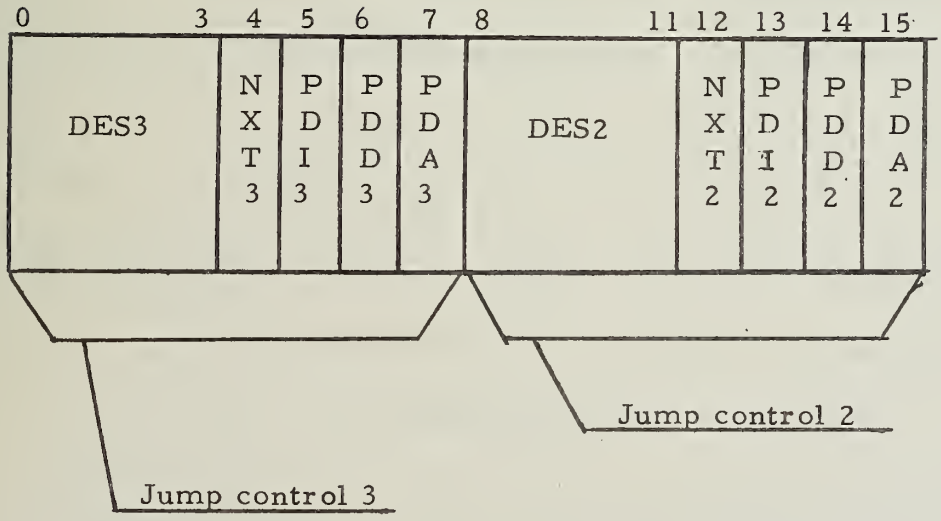


Figure 8 S4 word format

E. Decoding sequence

The string instruction explained in detail in V-A through V-D is decoded in the following way;

1. Parameters needed for the execution of an instruction are loaded into registers A1, A2, A3, CTR as needed in the way specified by the S1 word, the S2 word bits 7 through 15, and bits 11 through 15 of the S3 word. The parameters you need to set are explained under V-F.

2. Then according to the bits 0 through 6 of the S2 word, the instruction is executed and according to the result of the execution, program control is returned to Jump control 1 through 3, that are bits 0 through 10 of S3 word, bits 8 through 15 of S4 word and bit 0 through 7 of S4 word. When and where the control is returned are explained under V-F.

3. Then the pointers are changed as specified by Jump control bits of your return and the program control is passed to the instruction specified also by the Jump control bits of your return.

4. After every execution of string instructions, the difference of the data pointer position before the execution of the instruction and immediately before the execution of the instruction, which is after step 3, is placed in Counter Register 0 (CTR0).

F. String instruction specification

There are six string instructions which are explained next. In this computer system, one word consists of two characters and one character consists of eight bits. In order to make things easier, the following notations are used:

PD	Pointer to a string of data and placed in A3 register during the execution of instructions. We call this the data pointer.
PK	Second pointer when two pointers are involved and placed in A2 register. We call this the key pointer.
X	Arbitrary character without restriction on the position in a word.
X|	Arbitrary character placed in lower byte of a word.
|X	Arbitrary character placed in higher byte of a word.
	Pointer position

1. Search forward. . . Before the execution of this instruction, parameters must be placed as shown in Figure 9.

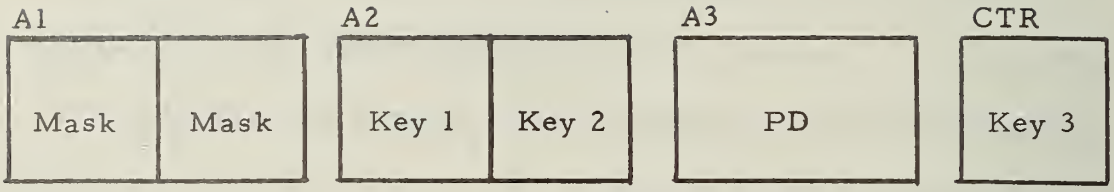


Figure 9 Parameter for search forward

This instruction looks for Key1, Key2, and Key3 in string PD. Before data is compared with Key1 or Key2, the data is ORed with MASK and then compared. There is no mask facility for Key3.

Initial state PD ↓

Pointer X X X - - - - - - - X X

Find Key1 state PD ↓

Pointer X X- - -X (X) X- - - (X)= Key1 masked

Decode exit Jump control 2

Find Key2 state PD ↓

Pointer X X- - -X (X) X- - - (X)= Key2 masked

Decode exit Jump control 3

Find Key3 state PD ↓

Pointer X X- - -X (X) X- - - (X)= Key3

Decode exit Jump control 1

2. Search reverse. . . Before the execution of this instruction, parameters must be placed as shown in Figure 10.

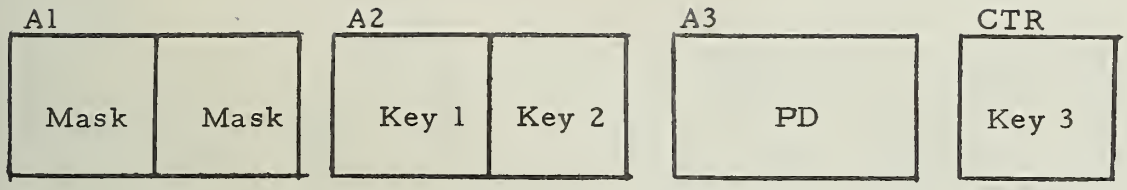


Figure 10 Parameter for search reverse

This instruction looks for Key1, Key2 and Key3 in string PD in reverse direction. Before data is compared with Key1 or Key2, the data is ORed with MASK and then compared. There is no mask facility for Key3.

Initial state

Pointer X X X X X X (PD)

Find Key1 state

Pointer X X- - - - X (X) X- - - -X (X)= Key1 masked

Decode exit Jump control 2

Find Key2 state

Pointer X X- - - -X (X) X- - - -X (X)= Key2 masked

Program control Jump control 3

Find Key3 state

Pointer X X- - - -X (X) X- - - -X (X)= Key3

Program control Jump control 1

3. Compare. . . Before the execution of this instruction, parameters must be placed as shown in Figure 11.

Figure 11 Parameter for compare

This instruction compares two strings PK and PD until it encounters unmatched character or Key1 in string PK.

Initial state

Pointer PD — PD ↓ : X X - - - - - -

Pointer PK — PK ↓ : X X - - - X (X) - - (X) = Key 1

Didn't compare state

Pointer PD — PD ↓ : [X] [X] - - - [X] X X - - X ([X] = compared character)

Pointer PK — PK ↓ : X X - - - - - - (X)

Decode exit — Jump control 2

Compared state

Pointer PD — PD ↓ : [X] [X] - - - [X] X

Pointer PK — PK ↓ : [X] [X] - - - [X] (X)

Decode exit — Jump control 1

4. Find. . . Before the execution of this instruction, parameters must be placed as shown in Figure 12.

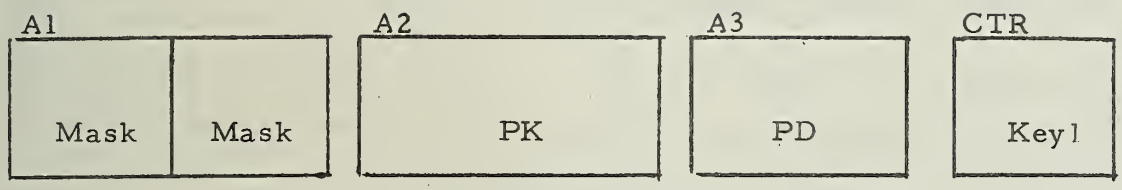


Figure 12 Parameter for find

This instruction looks for string PK in string PD until it encounters Key1 in string PD. When the first character in string PK is compared with data in string PD, the data is ORed with MASK and then compared. Other bits are not masked.

Initial state

Pointer PD — PD ↓ X X - - - - - - X (X) (X) = Key1

Pointer PK — PK ↓ X X- - - - - X (X)

Find state

Pointer PD — PD ↓ X X - -X [X] - - - [X] X X - -

Pointer PK — PK ↓ [X] [X]- -[X] (X) [X] = matched character

Didn't find state

Pointer PD — PD ↓ X X - - - - X X (X)

Pointer PK — PK ↓ X X - - - - X (X)

Decode exit — Jump control 2

5. Move forward. . . Before the execution of this instruction, parameters must be set as shown in Figure 13.

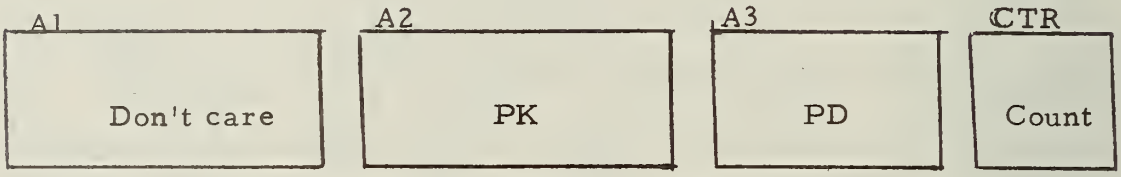
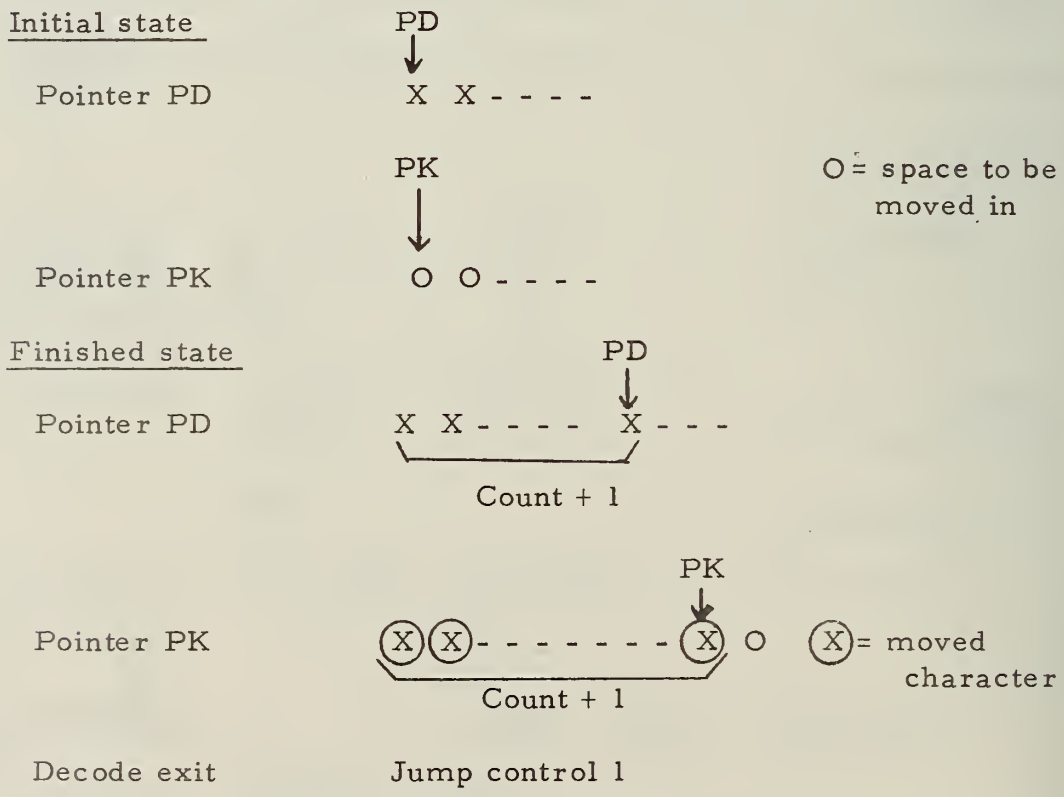


Figure 13 Parameters for move forward

This instruction moves (count +1) MOD 256 characters from string PD to string PK in the forward direction.



6. Move reverse. . . Before the execution of this instruction, parameters must be placed as shown in Figure 14.

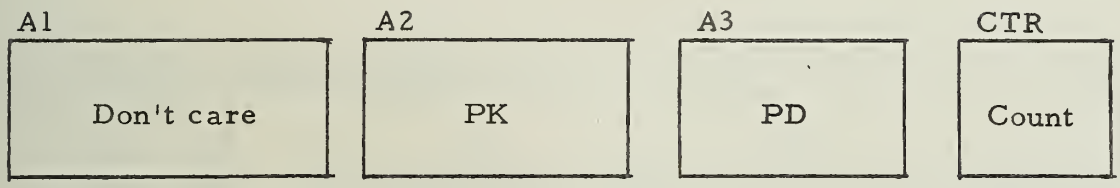
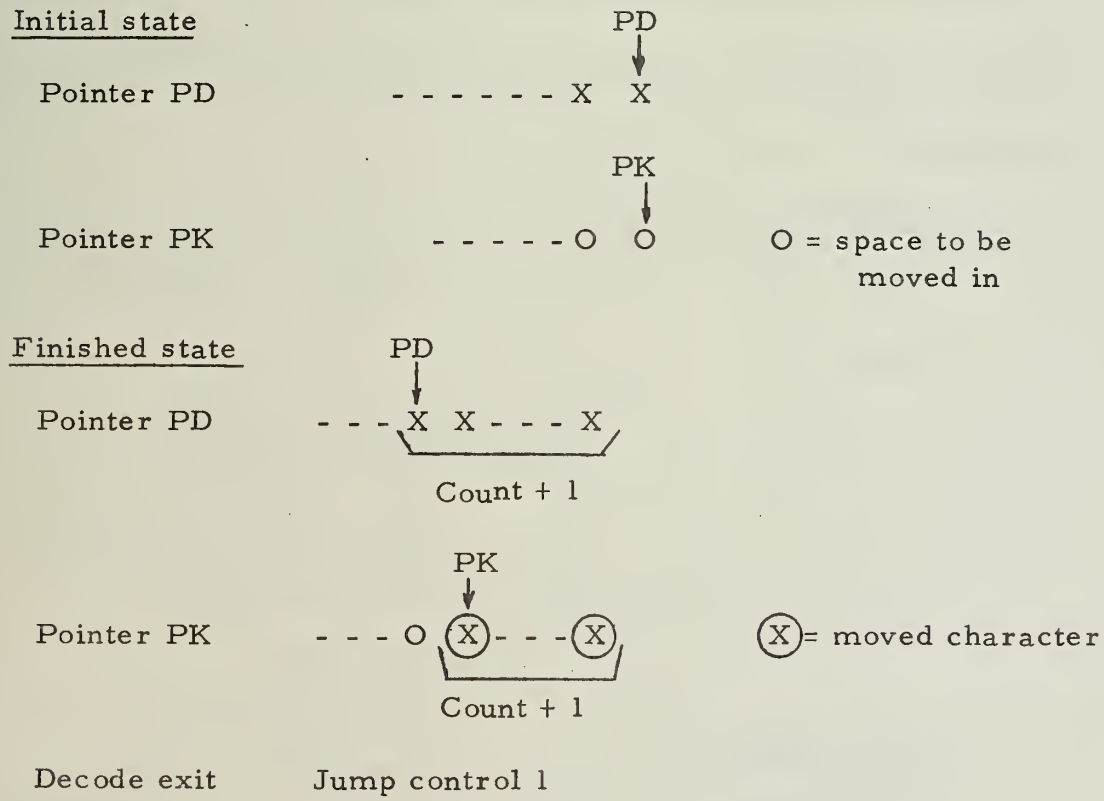


Figure 14 Parameters for move reverse

This instruction moves (count + 1) MOD 256 characters from string PD to string PK in reverse direction.



7. Translate. . . Before the execution of this instruction, parameters must be set as shown in Figure 15.

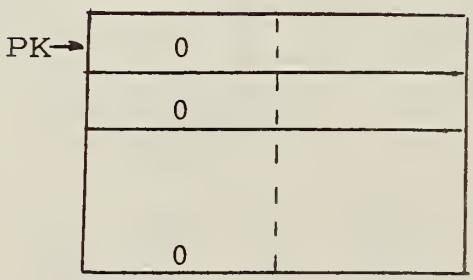
Figure 15 Parameters for translate

This instruction translates each character in string PD according to the translate table whose location is pointed to by PK and places the results in string PD. It continues translation until it encounters Key1 in string PD.

Initial state

Pointer PD

Pointer PK



Translation table

Finished state

Pointer PD

Pointer PK — unchanged

Decode control — Jump control 2

VI. Examples of String Instructions

Example 1. Find string PK in string PD. Pointer to the string PK is in PTR1 and pointer to the string PD is in PTR2. The ends of both strings are indicated by '+' mark which is in CHR1. If string PK is found in string PD, you want to jump to (IAR3). And you want the pointer PK at the position before the execution of this instruction and the pointer PD at the position after the execution. If string PK is not found, you want pointer PD and PK at the positions before the execution and want to go to the next instruction. This problem is visualized in Figure 16 and coding is shown in Figure 17.

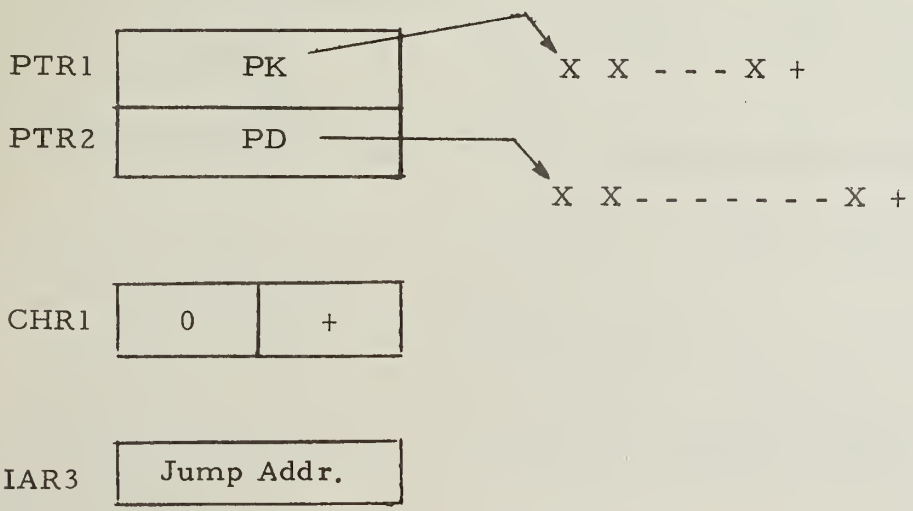
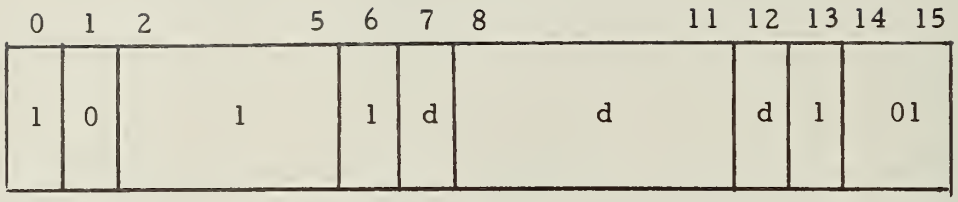
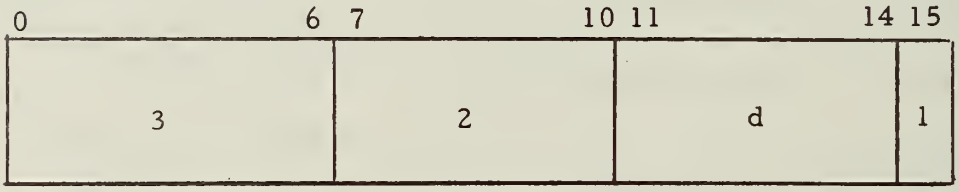


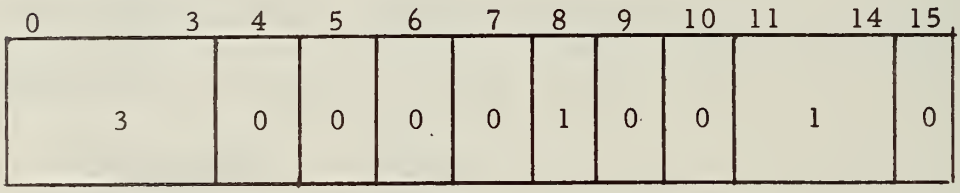
Figure 16 Picture of example 1



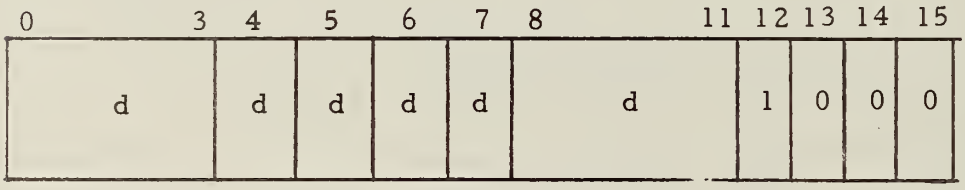
(a) S1 coding



(b) S2 coding



(c) S3 coding



(d) S4 coding

Figure 17 Coding for example 1

Example 2. You want to shift string PD which is pointed by PTR1 one place left until either '.' mark or '+' mark is encountered. Those marks are set in CHR5 and 6. This problem is solved by two string instructions. This problem is visualized in Figure 18.

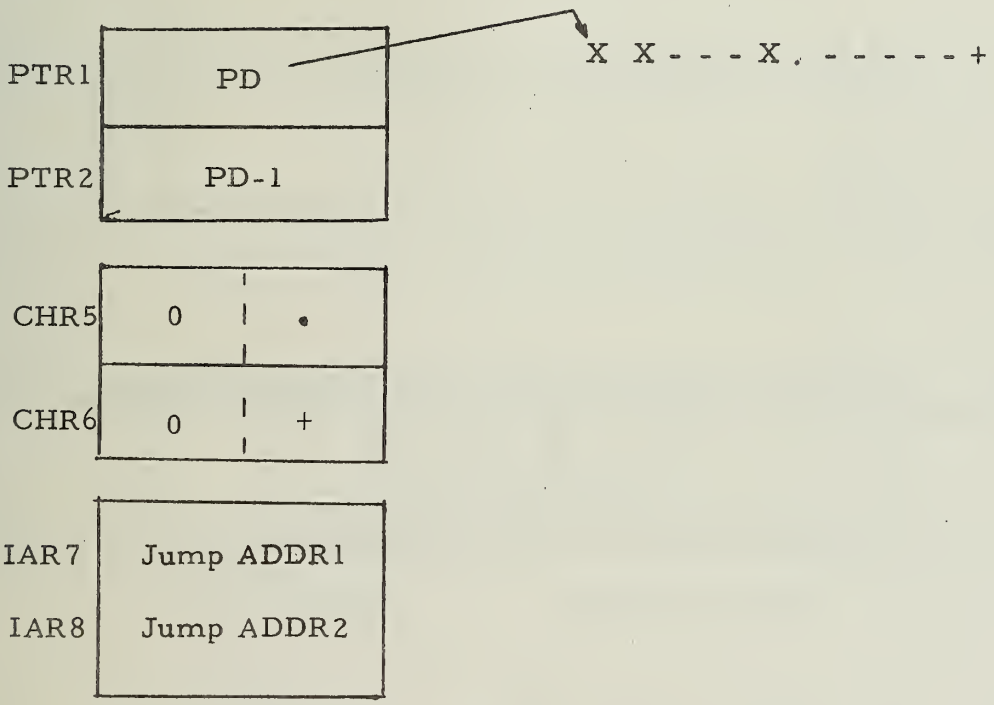
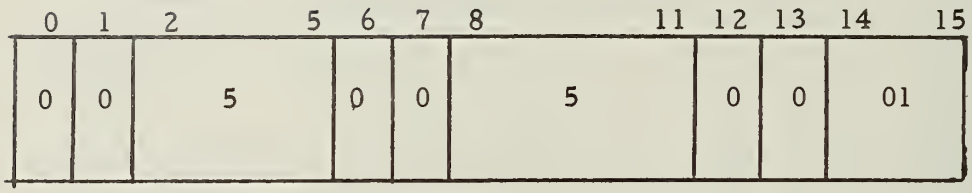


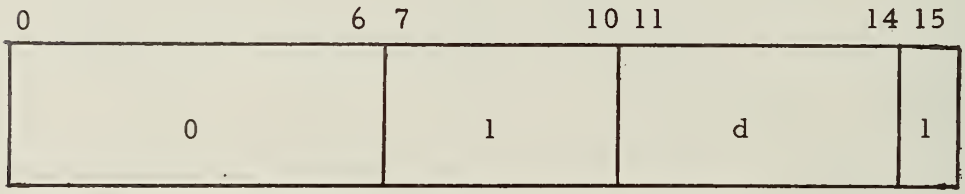
Figure 18 Picture of example 2

(1) Search for the character '+' or '.' in string PD, in order to know how many characters have to be moved. Coding of this instruction is shown in Figure 19.

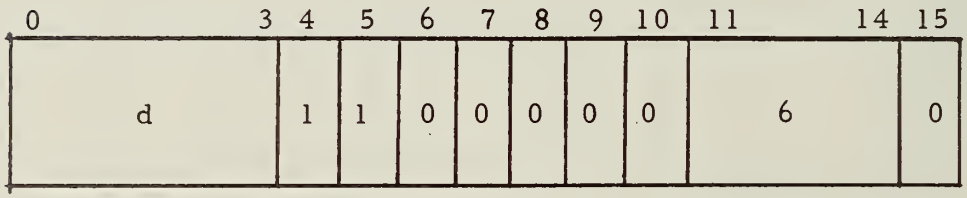
(2) After Step 1, CTR 0 contains the number of characters we have to move. So initiate move forward instructions using this register. This coding is shown in Figure 20.



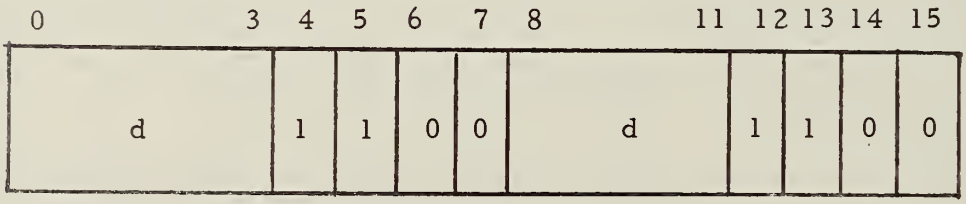
(a) S1 coding



(b) S2 coding

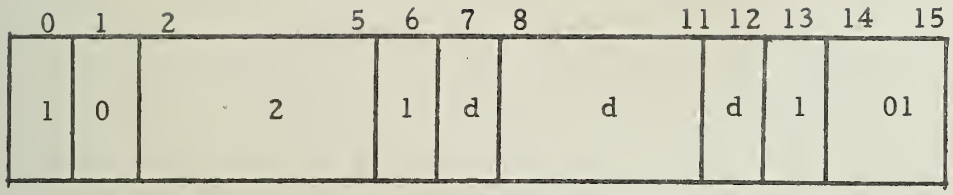


(c) S3 coding

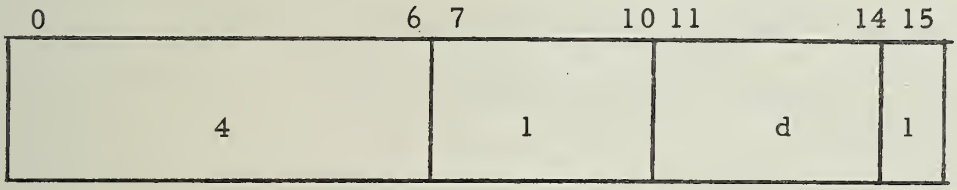


(d) S4 coding

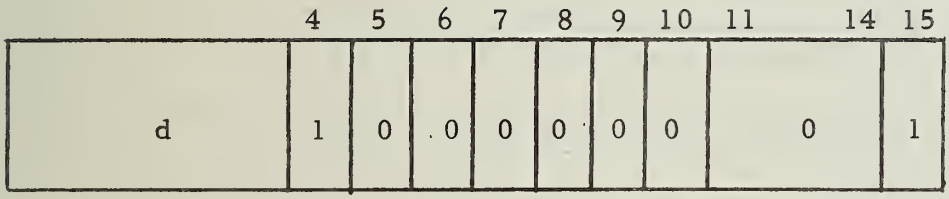
Figure 19 Coding of string instruction for example 2, step 1



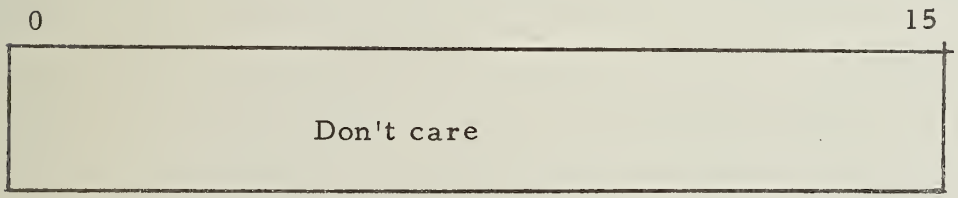
(a) S1 coding



(b) S2 coding



(c) S3 coding



(d) S4 coding

Figure 20 Coding of string instruction for example 2, step 2.

List of References

"APL/360 Primer," IBM program product, GH20-0689-1.

"APL/360 User's Manual," IBM program product, GH20-0683-1.

Bingham, H. W., "The BD: Machine an APL Model for Micro-Instruction Execution in Interpreter Based Systems," Burroughs Corporation, TR70-3, April 30, 1970.

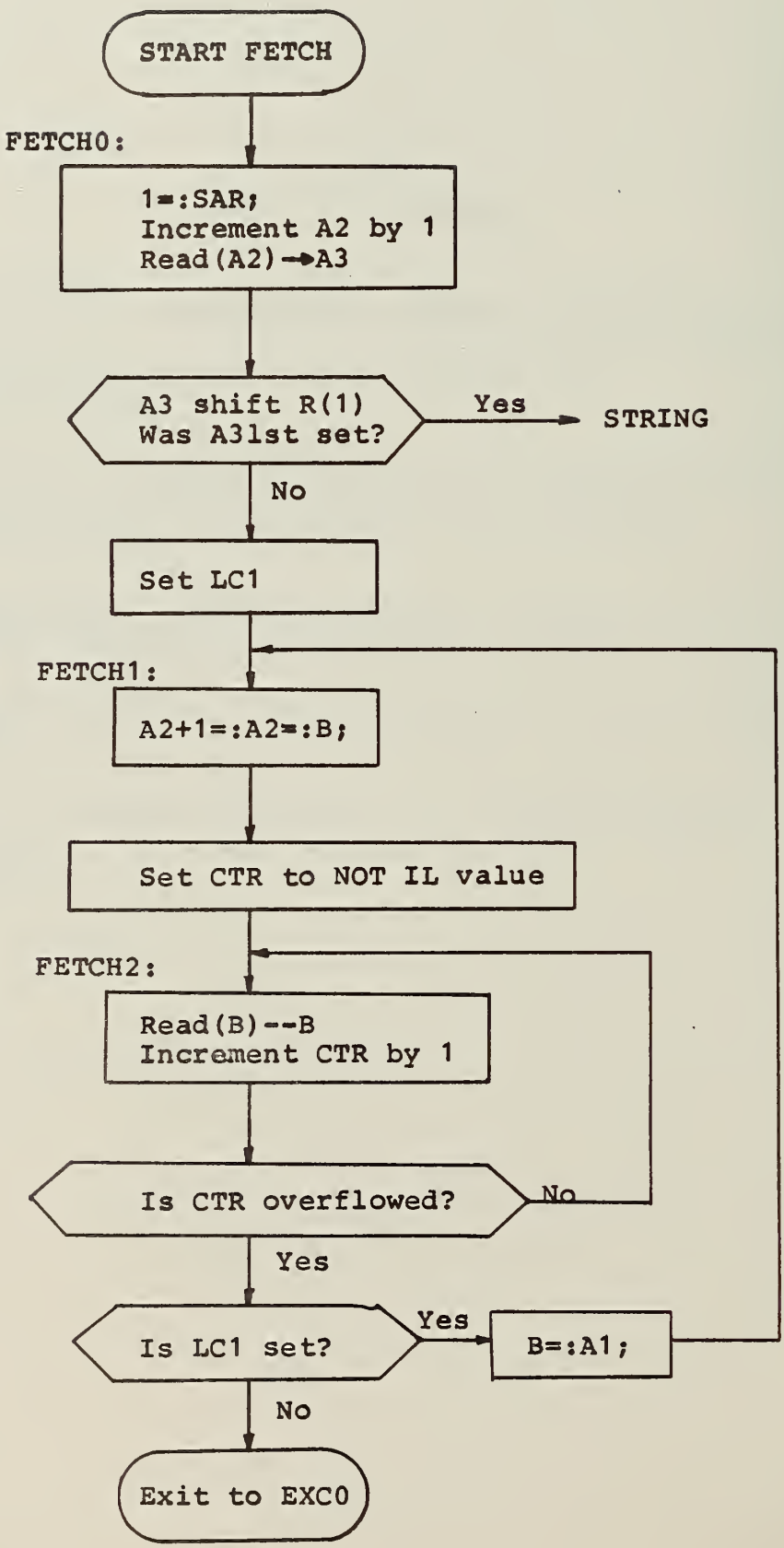
"Burroughs B5500 Time Sharing System--Terminal User's Guide," Burroughs Corporation.

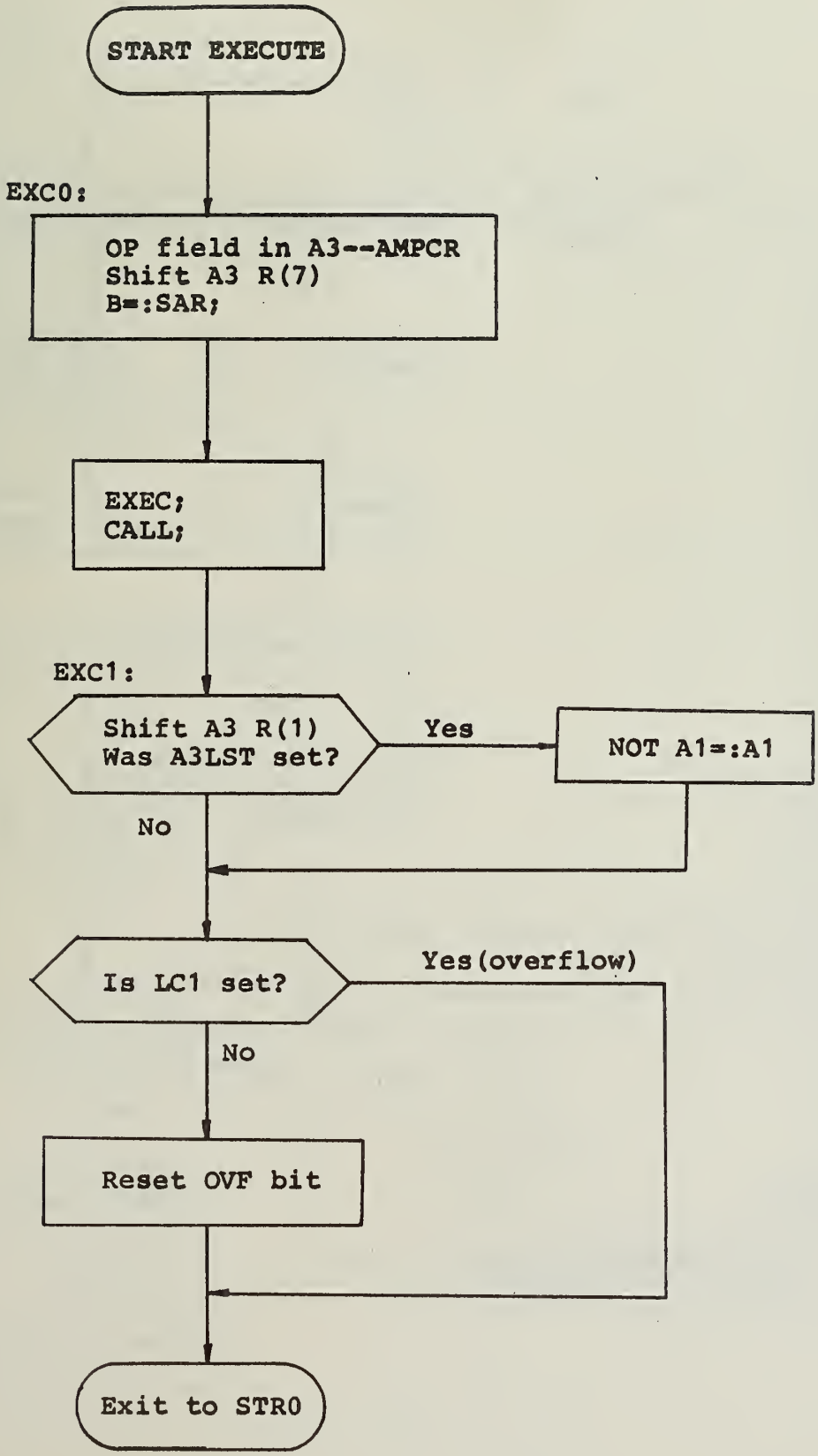
Burroughs Corporation, "The Interpreter," TR70-2, February 16, 1970.

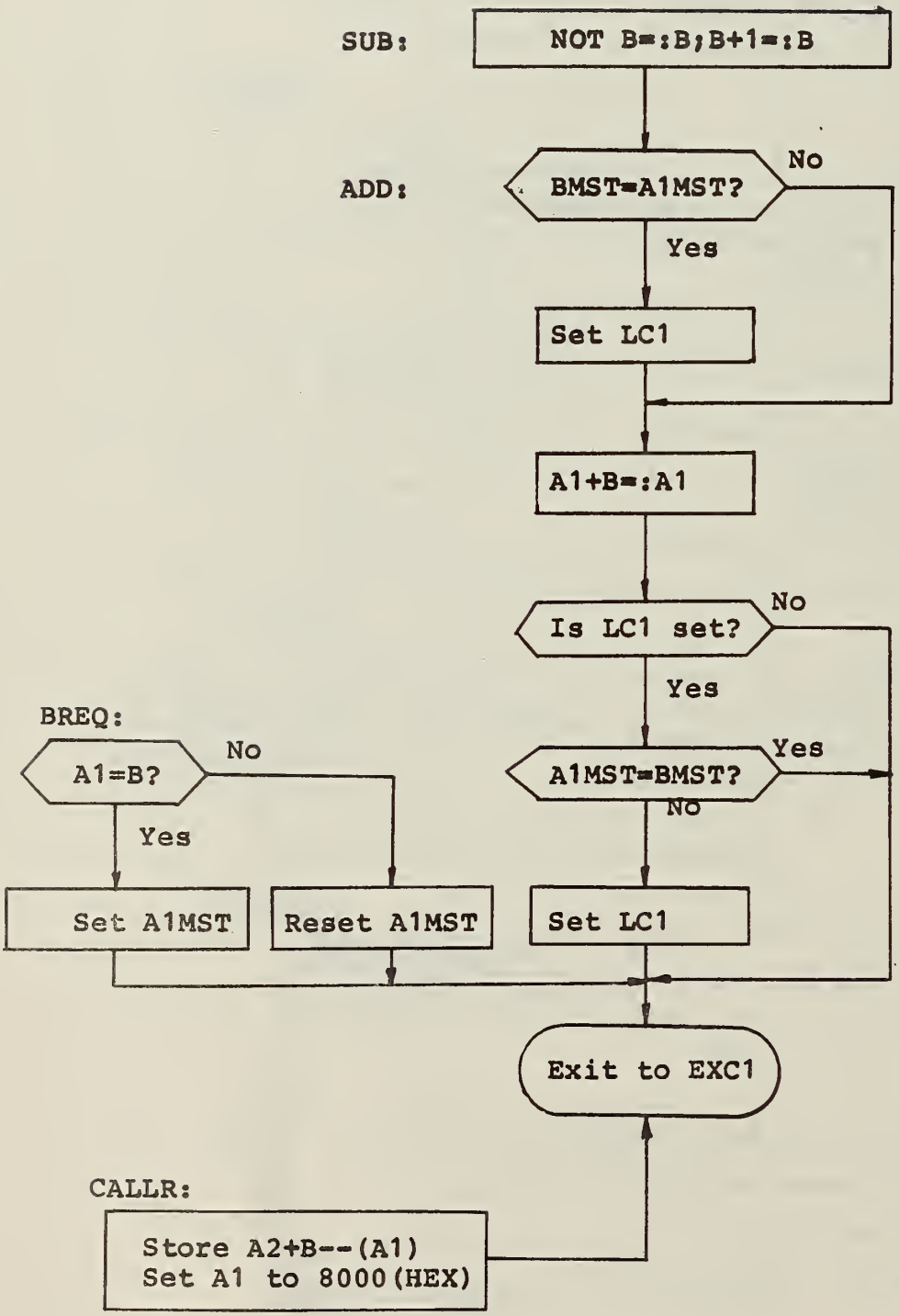
Zucker, S., "A Micro-Translator for the Interpreter Based Systems," Burroughs Corporation, TR70-1, February 10, 1970.

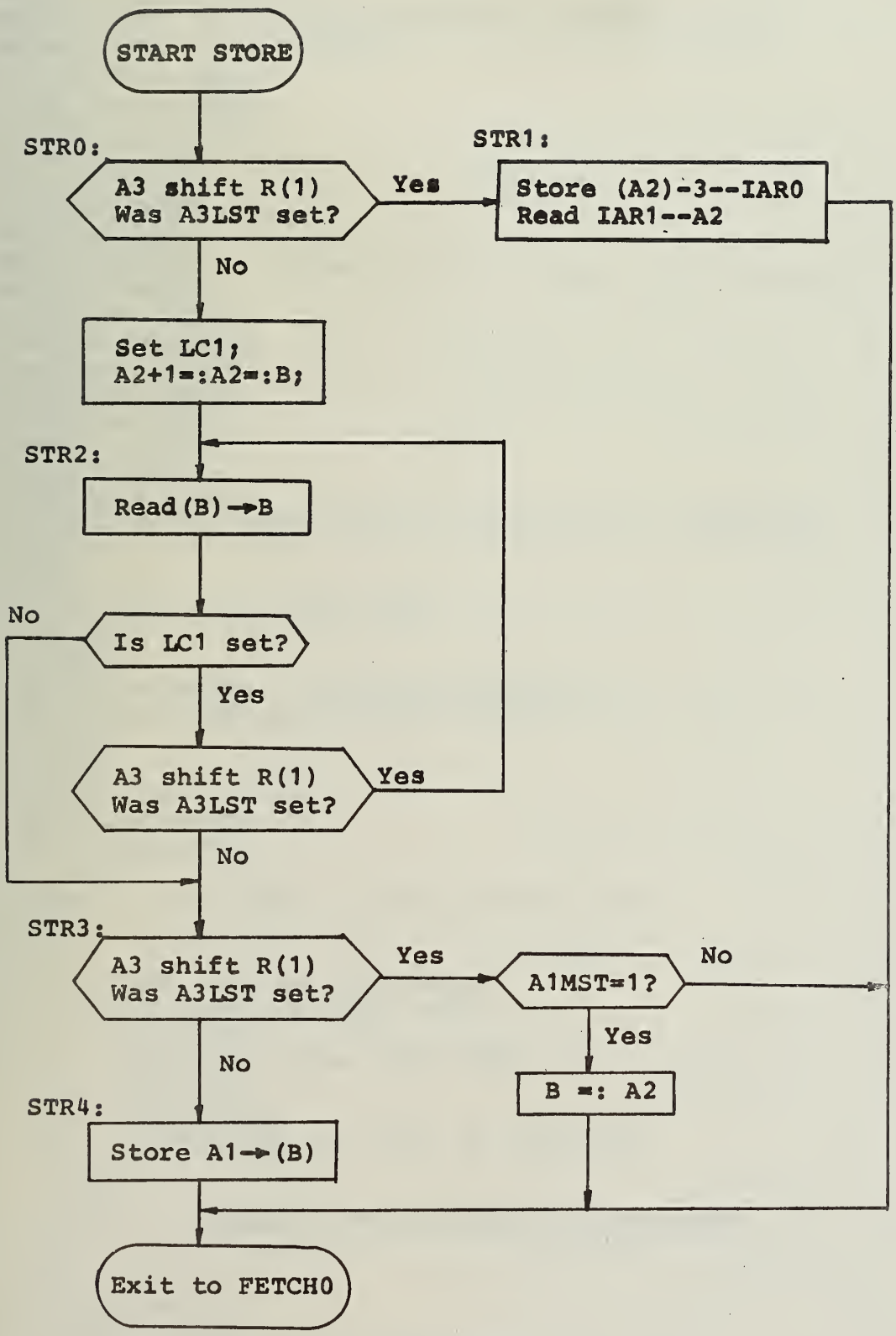
Appendix A

Flow chart of word instruction interpretation


START FETCH
FETCH0:
1=:SAR;
Increment A2 by 1
Read(A2)→A3
A3 shift R(1)
Was A31st set?
Yes
STRING
No
Set LC1
FETCH1:
A2+1=:A2=:B;
Set CTR to NOT IL value
FETCH2:
Read(B)--B
Increment CTR by 1
Is CTR overflowed?
No
Yes
Is LC1 set?
Yes
B=:A1;
No
Exit to EXC0


START EXECUTE
EXC0:
OP field in A3--AMPCR
Shift A3 R(7)
B=:SAR;
EXEC;
CALL;
EXC1:
Shift A3 R(1)
Was A3LST set?
Yes
NOT A1=:A1
No
Is LC1 set?
Yes(overflow)
No
Reset OVF bit
Exit to STR0


SUB:
NOT B=:B;B+1=:B
ADD:
BMST=A1MST?
No
Yes
Set LC1
A1+B=:A1
Is LC1 set?
No
Yes
BREQ:
A1=B?
No
Yes
A1MST=BMST?
Yes
No
Set A1MST
Reset A1MST
Set LC1
Exit to EXC1
CALLR:
Store A2+B--(A1)
Set A1 to 8000(HEX)


START STORE
STR0:
A3 shift R(1)
Was A3LST set?
Yes
STR1:
Store (A2)-3--IAR0
Read IAR1--A2
No
Set LC1;
A2+1=:A2=:B;
STR2:
Read(B)→B
No
Is LC1 set?
Yes
A3 shift R(1)
Was A3LST set?
Yes
No
STR3:
A3 shift R(1)
Was A3LST set?
Yes
A1MST=1?
No
Yes
B =: A2
No
STR4:
Store A1→(B)
Exit to FETCH0

Appendix B

Flow chart of word instruction interpretation

```
50 PROGRAM WRDINS ADR 1;
60 COMMENT THIS IS THE INTERPRETATION PROGRAM OF WORD
61 INSTRUCTIONS FOR D PROJECT;
100 A1 AND B=:A1;SKIP;
200 A1 OR B=:A1;SKIP;
300 A1 EQV B=:A1;SKIP;
400 A1 R=:A1;SKIP;
500 A1 L=:A1;SKIP;
600 A1 C=:A1;SKIP;
700 PLUS-1=:AMPCR;
800 SUB-1=:AMPCR;
900 BREQ-1=:AMPCR;
1000 CALLR-1=:AMPCR;
1100 SRCHF0-1=:AMPCR;
1200 SRCHR0-1=:AMPCR;
1300 CMPR0-1=:AMPCR;
1400 FIND0-1=:AMPCR;
1500 MVF0-1=:AMPCR;
1600 MVR0-1=:AMPCR;
1700 TRNSL0-1=:AMPCR;
2000 FETCH0: IF RMA THEN A2+B001=:BR2=:MAR=:A2 ELSE WAIT;
2050 STEP;
2100 MR2;
2200 IF RDC THEN ,BEX ELSE WAIT;
2300 B R=:A3;
2350 1=:SAR;
2400 STRING-1=:AMPCR;   %STRING INSTRUCTION
2500 IF NOT LST THEN SET LC1 ELSE JUMP;
2600 2=:SAR;
2650 3=:LIT;
2800 FETCH1: A2+1=:B=:A2;
2900 FETCH2-1=:AMPCR;
3000 A3 AND LIT=:CTR;
3100 A3 R=:A3;
3200 FETCH2: IF RMA THEN B=:BR2=:MAR ELSE WAIT;
3250 STEP;
3300 MR2;
3400 IF RDC THEN ,BEX,INC ELSE WAIT;
3500 IF NOT COV THEN JUMP ELSE STEP;
3600 FETCH1-1=:AMPCR;
3700 IF LC1 THEN B=:A1;JUMP ELSE STEP;
3900 EXC0: 127=:LIT;
4000 A3 AND LIT=:AMPCR;
4050 STEP;   %DUMMY DUE TO A BUG IN SIMULATOR
4100 7=:SAR;
4200 A3 R=:A3;
4300 B=:SAR;     %PREPARE IN CASE SHIFT INSTRUCTION
4400 EXEC;
4500 CALL;
4600 EXC1: A3 R=:A3;
4700 1=:SAR;
4800 IF LST THEN NOT A1=:A1 ELSE STEP;
4900 IF NOT LC1 THEN A3 AND B110=:A3 ELSE STEP;
```

```
5000 STR0: A3 R=:A3;
5100 STR1-1=:AMPCR;
5200 IF NOT LST THEN A2+B001=:A2=:B;SET LC1 ELSE JUMP;
5300 STR2: IF RMA THEN B=:BR2=:MAR ELSE WAIT;
5350 STEP;
5400 MR2;
5500 STR3-1=:AMPCR;
5600 IF RDC THEN ,BEX ELSE WAIT;
5700 IF LC1 THEN A3 R=:A3 ELSE JUMP;
5800 STR2-1=:AMPCR;
5900 IF LST THEN JUMP ELSE STEP;
6000 STR3: A3 R=:A3;
6100 STR4-1=:AMPCR;
6200 IF LST THEN A1 ELSE JUMP;
6300 FETCH0-1=:AMPCR;
6400 IF MST THEN B=:A2;JUMP ELSE JUMP;
6500 STR4: IF RMA THEN B=:BR2=:MAR ELSE WAIT;
6600 IF RMI THEN A1=:MIR ELSE WAIT;
6700 FETCH0-1=:AMPCR;
6750 STEP;
6800 MW2;JUMP;
6900 STR1: 16=:LIT;
7000 IF RMA THEN ,LMAR ELSE WAIT;
7100 NOT LIT=:B;
7200 3=:LIT;
7300 IF RMI THEN A2 ADL B=:MIR ELSE WAIT;
7350 STEP;
7400 MW1;
7500 IF RMA THEN ,LMAR ELSE WAIT;
7600 17=:LIT;    %ADDR IAR1 IS 17
7650 STEP;
7700 MR1;
7800 FETCH0-1=:AMPCR;
7900 IF RDC THEN ,BEX ELSE WAIT;
8000 B=:A2;JUMP;
8100 SUB: NOT B=:B;
8200 ADL B=:B;
8300 PLUS: A1 EQV B;
8400 IF MST THEN SET LC1 ELSE STEP;
8500 A1+B=:A1;
8600 EXC1-1=:AMPCR;
8700 IF LC1 THEN A1 EQV B ELSE JUMP;
8800 IF NOT MST THEN SET LC1;JUMP ELSE JUMP;
8900 BREQ: A1 EQV B=:B;
9000 0=:A1;
9100 ADL B=:B;
9200 EXC1-1=:AMPCR;
9300 IF AOV THEN A1 OR B100=:A1;JUMP ELSE JUMP;
9400 CALLR: IF RMA THEN A1=:BR2=:MAR ELSE WAIT;
9500 EXC1-1=:AMPCR;
9600 IF RMI THEN A2+B=:MIR ELSE WAIT;
9650 STEP;
9700 MW2;B100=:A1;JUMP;
9800 END
```